CLAIMS

What is claimed is:

1. A method for forming and executing code units, comprising:

forming a first code unit having a resource, wherein the first code unit places restrictions on the visibility of the resource to other code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to a second code unit;

forming the second code unit; and

executing the first and second code units using a runtime component, involving making the resource of the first code unit visible to the second code unit as instructed by the attribute,

wherein the executing comprises using the attribute to determine, at runtime, whether the second code unit has permission to view the resource of the first code unit.

2. The method of claim 1, wherein the attribute that overrides the restrictions comprises a custom attribute added to the first code unit.

3. The method of claim 1, wherein the attribute identifies a name of the second code unit.

4. The method of claim 1, wherein the attribute identifies a security code associated with the second code unit.

5. The method of claim 4, wherein the security code is a public key token.

6. The method of claim 1, wherein the attribute identifies a version of the second code unit.

7. The method of claim 6, wherein the forming of the second code unit and/or the executing applies fuzzy matching to determine whether a version of the second code unit that is being processed matches the version identified in the attribute of the first code unit, the fuzzy matching permitting similar versions to constitute a match that are not exactly the same.

8. The method of claim 1, wherein the attribute identifies culture information associated with the second code unit.

9. The method of claim 1, wherein the forming of the first code unit includes compiling source code that includes the attribute contained therein.

10. The method of claim 1, wherein the forming of the first code unit includes specifying at least some information associated with the attribute via a command line operation.

11. The method of claim 1, wherein the attribute applies to the entire first code unit, making all restricted resources in the first code unit available to the second code unit.

12. The method of claim 1, wherein the attribute applies locally to the resource.

13. The method of claim 1, wherein the resource is one of: an assembly, a module within an assembly, a type, and a member.

14. The method of claim 1, wherein the forming of the second code unit involves ensuring that the second code unit can view the resource.

15. The method of claim 1, wherein the forming of the second code unit does not involve ensuring that the second code unit can view the resource.

16. The method of claim 1, wherein the first and second code units are formed as managed code that targets a virtual machine processing environment.

17. The method of claim 16, wherein the first and second code units are formed to exploit Common Language Runtime functionality provided by a .NET framework.

18. The method of claim 1, wherein the first and second code units can be formed using any one of multiple different code languages, and the determination of whether the second code unit has permission to view the resource of the first code unit is performed at runtime in a manner which is independent of the code language used.

19. A computer readable medium including machine readable instructions for implementing each of the forming of the first code unit, forming of the second code unit, and executing of the first and second code units of claim 1.

20. A method for executing code units, comprising:

receiving first and second compiled code units, wherein the first code unit places restrictions on the visibility of a resource in the first code unit to other code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to the second code unit;

using the attribute to determine, at runtime, whether the second code unit has permission to view the resource of the first code unit; and

if it is determined that the second code unit has permission to view the resource of the first code unit, then proceeding with the execution of the second code unit.

21. The method of claim 20, wherein the attribute also identifies a security code associated with the second code unit, and wherein the determining further comprises determining whether a security code associated with the second code unit matches the security code identified by the attribute, and the proceeding with the execution is performed only on condition of a match.

22. The method of claim 20, wherein the first and second code units can be formed using any one of multiple different code languages, and the determination of whether the second code unit has permission to view the resource of the first code unit is performed at runtime in a manner which is independent of the code language used.

23. A computer readable medium including machine readable instructions for implementing each of the receiving, determining, and proceeding of claim 20.

24. A system for forming and executing code units, comprising:

a language compiler configured to:

form a first code unit having a resource, wherein the first code unit places restrictions on the visibility of the resource to other code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to a second code unit; and

form the second code unit; and

a runtime component configured to execute the first and second code units using a runtime compiler, involving making the resource of the first code unit visible to the second code unit as instructed by the attribute,

wherein the runtime component is configured to use the attribute to determine, at runtime, whether the second code unit has permission to view the resource of the first code unit.

25. The system of claim 24, wherein the attribute that overrides the restrictions comprises a custom attribute added to the first code unit.

26. The system of claim 24, wherein the attribute identifies a name of the second code unit.

27. The system of claim 24, wherein the attribute identifies a security code associated with the second code unit.

28. The system of claim 27, wherein the security code is a public key token.

29. The system of claim 24, wherein the attribute identifies a version of the second code unit.

30. The system of claim 29, wherein the language compiler and/or the runtime component is configured to apply fuzzy matching to determine whether a version of the second code unit that is being processed matches the version identified in the attribute of the first code unit, the fuzzy matching permitting similar versions to constitute a match that are not exactly the same.

31. The system of claim 24, wherein the attribute identifies culture information associated with the second code unit.

32. The system of claim 24, wherein the language compiler is configured to form the first code unit by compiling source code that includes the attribute contained therein.

33. The system of claim 24, wherein the language compiler is configured to form the first code unit by specifying at least some information associated with the attribute via a command line operation.

34. The system of claim 24, wherein the attribute applies to the entire first code unit, making all restricted resources in the first code unit available to the second code unit.

35. The system of claim 24, wherein the attribute applies locally to the resource.

36. The system of claim 24, wherein the resource is one of: an assembly, a module within an assembly, a type, and a member.

37. The system of claim 24, wherein the language compiler is configured to form the second code unit by ensuring that the second code unit can view the resource.

38. The system of claim 24, wherein the language compiler is configured to form of the second code unit by not ensuring that the second code unit can view the resource.

39. The system of claim 24, wherein the first and second code units are formed as managed code that targets a virtual machine processing environment.

40. The system of claim 39, wherein the first and second code units are formed to exploit a Common Language Runtime provided by a .NET Framework.

41. The system of claim 24, wherein the first and second code units can be formed using any one of multiple different code languages, and wherein the runtime component is configured to use the attribute to determine whether the second code unit has permission to view the resource of the first code unit in a manner which is independent of the code language used.

42. A computer readable medium including machine readable instructions for implementing each of the language compiler and the runtime component of claim 24.

43. A runtime component for executing code units, comprising:

logic configured to receive first and second compiled code units, wherein the first code unit places restrictions on the visibility of a resource in the first code unit to other

code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to the second code unit;

logic configured to use the attribute to determine whether the second code unit has permission to view the resource of the first code unit; and

logic configured to proceed with the execution of the second code unit if it is determined that the second code unit has permission to view the resource of the first code unit.

44. The runtime component of claim 43, wherein the attribute also identifies a security code associated with the second code unit, and wherein the logic for determining is further configured to determine whether a security code associated with the second code unit matches the security code identified by the attribute, and the logic for proceeding with the execution is configured to execute the second code unit only on condition of a match.

45. The runtime component of claim 43, wherein the first and second code units can be formed using any one of multiple different code languages, and wherein the logic configured to determine whether the second code unit has permission to view the resource of the first code unit is configured to perform the determination in a manner which is independent of the code language used.

46. A computer readable medium including machine readable instructions for implementing the logic for receiving, logic for determining, and logic for proceeding of claim 43.

47. A method for compiling code units, comprising:

forming a first code unit having a resource, wherein the first code unit places restrictions on the visibility of the resource to other code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to a second code unit; and

forming the second code unit, including ensuring that the second code unit can view the resource in the first code unit,

wherein the attribute is interpretable at runtime to selectively permit the second code unit to access the resource in the first code unit, and wherein the interpretation of the attribute is performed in a manner which is independent of a code language used to form the first code unit and the second code unit.

48. A computer readable medium including machine readable instructions for implementing each of the forming of the first code unit, and forming of the second code unit of claim 47.

49. A language compiler for forming code units, comprising:

logic configured to:

form a first code unit having a resource, wherein the first code unit places restrictions on the visibility of the resource to other code units, and wherein the first code unit includes an attribute which overrides the restrictions with respect to a second code unit, and ; and

form the second code unit,

wherein the logic is further configured to ensure that the second code unit can view the resource in the first code unit when it forms the second code unit,

further wherein the attribute is interpretable at runtime to selectively permit the second code unit to access the resource in the first code unit, and wherein the interpretation of the attribute is performed in a manner which is independent of a code language used to form the first code unit and the second code unit.

50. A computer readable medium including machine readable instructions for implementing the logic of claim 49.